

情報理工学部 SN コース 2 回  
セキュリティ・ネットワーク学実験 2  
課題 5-3 レポート

2600200443-6  
Yamashita Kyohei  
山下 恭平

December 17 2021

## 1 概要

接続許可リストにあるクライアントのみに対し、クライアントから送られてきた文字列を変換し、クライアントへ返送する TCP サーバプログラムを作成する。

## 2 外部仕様

### 2.1 サーバ側

サーバは起動と同時に図 1 の状態になり、クライアントからの接続を待っている状態となる。許可した IP アドレスから接続が行われた時「conected :接続先 IP アドレス」と表示され、許可されていない IP アドレスからの接続が行われた場合は、その IP アドレスを表示し、「許可されていない IP アドレスからのアクセス」と表示されるようになっている。その様子を図 2,3 に示す。

```
[kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_server
```

図 1 接続待ち状態

```
[kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_server  
conected :127.0.0.1
```

図 2 許可した相手からの接続


```
[kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_server  
127.0.0.1  
許可されていないIPアドレスからのアクセス
```

図 3 許可していない相手からの接続

### 2.2 クライアント側

クライアント側は図 4 のように、起動時、第二引数にホスト名、第三引数にポート番号を指定することで接続先を任意に指定することができる。自身の IP アドレスがサーバへの接続を許可されていた場合図 5 のように接続先の IP アドレスを表示し、送信するメッセージをキーボードから受け付けるようになっている。もし、自身の IP アドレスがサーバの許可を得ていない場合、図 6 のように接続が拒否された趣旨のメッセージを表示するようになっている。

```
[kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_client localhost 12345]
```



ホスト名                      ポート番号

図4 接続方法

```
[kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_client localhost 12345  
接続先 ip address : 127.0.0.1  
connect  
message >]
```

図5 接続した時

```
[kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_client localhost 12345  
接続先 ip address : 127.0.0.1  
connection refused
```

図6 接続が許可されていない場合

## 3 内部仕様

### 3.1 サーバ側

サーバは「acceptlist.txt」に書き込まれている IP アドレスを fgets で一行ずつ読み込み、接続先の IP アドレスと比較していき、もし一致するものがあれば、接続先の IP アドレスを表示、「threadfunc」を呼び出し、クライアント側の入力を待機、送られてきた文字列を大文字に変換し、送信し返すようになっている。もし、一致する IP アドレスが存在しなかった場合、接続先の IP アドレスと「許可されていない IP アドレスからのアクセス」を表示し、「refusefunc」を呼び出し、buf に「refused」と書き込み、それを送信するようにしている。以下は、「acceptlist.txt」の例である。

Listing 1 acceptlist

1	192.168.11.12
2	192.168.11.14
3	127.0.0.1

### 3.2 クライアント側

クライアント側はコマンドライン引数の第二引数にホスト名、第三引数にポート番号が格納されており、その情報をもとにソケットを生成し、サーバに接続する。サーバに接続後、サーバからのパケットを待機する。もし初めに送信されてきたデータに「refused」と書き込まれていた場合、自身の IP アドレスはサーバへの接続は許可されていないので「この IP アドレスからの接続は許可されていません」と表示し、通信を終了する。「refused」と書き込まれていない場合、送信するメッセージをキーボードから受け付け、それをサーバへ送信し、変換された文字を受信、その文字を表示し通信を終了する。

## 4 実行例

今回の実験では実験室に設置されているコンピュータをクライアント、自身のコンピュータをサーバとし、実験を行った。まず初めに自席に設置されているコンピュータの IP アドレスは「172.27.74.63」であったので、この IP アドレスを「acceptlist.txt」に書き込み、サーバを建て、クライアントから接続を行なったものが実行例 1 である。以下はその実際の様子を示したものである。

```
[koyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_server  
connected :172.27.74.63
```

図 7 実行例 1 サーバ

```
is0585xf@PR6IS063:~/ダウンロード  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
[is0585xf@PR6IS063 ダウンロード]$ gcc -o 5-3_client 5-3_client.c  
[is0585xf@PR6IS063 ダウンロード]$ ./5-3_client 172.31.145.35 12345  
接続先 ip address : 172.31.145.35  
connect  
message >00MOJIKomoji  
oomojiKOMOJI  
[is0585xf@PR6IS063 ダウンロード]$
```

図 8 実行例 1 クライアント

実行例 2 では、逆に「acceptlist.txt」から自席のコンピュータの IP アドレスを削除し、サーバを建て実行したものである。実行結果は以下の図のとおりであり、仕様通り、サーバ側では接続先の IP アドレスがサーバに表示され、許可されていないとの趣旨が表示され、クライアント側では接続が拒否されたことを示す内容を画面に表示している。

```
[koyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./5-3_server  
172.27.74.63  
許可されていないIPアドレスからのアクセス
```

図 9 実行例 2 サーバ

```
is0585xf@PR6IS063:~/ダウンロード  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
[is0585xf@PR6IS063 ダウンロード]$ gcc -o 5-3_client 5-3_client.c  
[is0585xf@PR6IS063 ダウンロード]$ ./5-3_client 172.31.145.35 12345  
接続先 ip address : 172.31.145.35  
connect  
message >00MOJIKomoji  
oomojiKOMOJI  
[is0585xf@PR6IS063 ダウンロード]$ ./5-3_client 172.31.145.35 12345  
接続先 ip address : 172.31.145.35  
connection refused  
[is0585xf@PR6IS063 ダウンロード]$
```

図 10 実行例 2 クライアント

実行例 3 は複数クライアントからの接続を行う実験である。「acceptlist.txt」には自身のコンピュータの IP アドレス「192.168.11.14」を書き込み、ローカルホストである「127.0.0.1」は書き込まなかった。実行結果

は、以下の図のとおりとなり、マルチクライアント環境においても、IP アドレスによる判別ができていことがわかる。

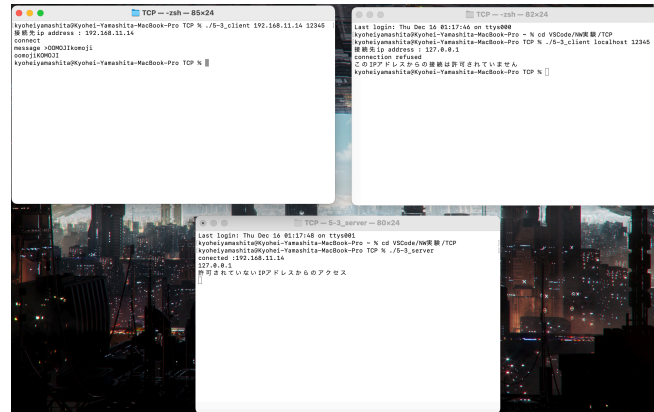


図 11 実行例 3 マルチキャスト

## 5 ソースコード

### 5.1 サーバ側

Listing 2 server

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <sys/types.h>
5  #include <sys/socket.h>
6  #include <netinet/in.h>
7  #include <arpa/inet.h>
8  #include <pthread.h>
9  #include <errno.h>
10 #include <string.h>
11
12 struct clientdata
13 {
14     int sock;
15     struct sockaddr_in saddr;
16 };
17
18 void *threadfunc(void *data)
19 {
20     int sock;
21     struct clientdata *cdata = data;
22     char buf[1024];
23     int n, i, errocheck;
24
```

```

25     if (data == NULL)
26     {
27         return (void *)-1;
28     }
29
30     /* (5)新規TCPコネクションのソケットを取得 */
31     sock = cdata->sock;
32
33     strcpy(buf, "connect");
34
35     errocheck = write(sock, buf, sizeof(buf));
36     if (errocheck < 0)
37     {
38         perror("write");
39         goto err;
40     }
41
42     n = read(sock, buf, sizeof(buf));
43     if (n < 0)
44     {
45         perror("read");
46         goto err;
47     }
48
49     for (i = 0; buf[i] != '\0'; i++)
50     {
51         if ('a' <= buf[i] && buf[i] <= 'z')
52         {
53             buf[i] = buf[i] - ('a' - 'A');
54         }
55         else
56         {
57             buf[i] = buf[i] + ('a' - 'A');
58         }
59     }
60
61     buf[i - 1] = '\0';
62
63     n = write(sock, buf, n);
64     if (n < 0)
65     {
66         perror("write");
67         goto err;
68     }
69
70     /* 新規TCPセッションの終了 */
71     if (close(sock) != 0)

```

```

72     {
73         perror("close");
74         goto err;
75     }
76     /* 親スレッドでmallocされた領域を開放 */
77     free(data);
78
79     return NULL;
80
81     err:
82     free(data);
83     return (void *)-1;
84 }
85
86 void *refusefunc(void *data)
87 {
88     int sock;
89     struct clientdata *cdata = data;
90     char buf[1024];
91     int n, i;
92
93     if (data == NULL)
94     {
95         return (void *)-1;
96     }
97
98     /* (5)新規TCPコネクションのソケットを取得 */
99     sock = cdata->sock;
100
101     strcpy(buf, "refused");
102
103     n = write(sock, buf, sizeof(buf));
104     if (n < 0)
105     {
106         perror("write");
107         goto err;
108     }
109
110     /* 新規TCPセッションの終了 */
111     if (close(sock) != 0)
112     {
113         perror("close");
114         goto err;
115     }
116     /* 親スレッドでmallocされた領域を開放 */
117     free(data);
118

```

```

119         return NULL;
120
121     err:
122         free(data);
123         return (void *)-1;
124     }
125
126     int main()
127     {
128         int sock0;
129         struct sockaddr_in addr;
130         socklen_t len;
131         pthread_t th;
132         struct clientdata *cdata;
133
134         FILE *fp; // FILE型構造体
135         char fname[] = "acceptlist.txt";
136         char str[256];
137
138         int refusecheck;
139
140         /* (1) */
141         /* ソケットの作成 */
142         sock0 = socket(AF_INET, SOCK_STREAM, 0);
143
144         /* ソケットの設定 */
145         addr.sin_family = AF_INET;
146         addr.sin_port = htons(12345);
147         addr.sin_addr.s_addr = INADDR_ANY;
148
149         bind(sock0, (struct sockaddr *)&addr, sizeof(addr));
150
151         /* TCPクライアントからの接続要求を待てる状態にする */
152         listen(sock0, 5);
153         /* (1) 終わり */
154
155         /* (2)新規TCPコネクションに関する情報をclientdata構造体に格納 */
156         for (;;)
157         {
158             refusecheck = 0;
159
160             cdata = malloc(sizeof(struct clientdata));
161             if (cdata == NULL)
162             {
163                 perror("malloc");
164                 return 1;
165             }

```



```

166
167      /* TCPクライアントからの接続要求を受け付ける */
168      len = sizeof(cdata->saddr);
169      cdata->sock = accept(sock0, (struct sockaddr *)&cdata->saddr, &len);
170
171      fp = fopen(fname, "r"); // ファイルを開く。失敗するとNULLを返す。
172      if (fp == NULL)
173      {
174          printf("%s file not open!\n", fname);
175          return -1;
176      }
177
178      while (fgets(str, 256, fp) != NULL)
179      {
180          int l = 0;
181
182          while (1)
183          {
184              if (str[l] == '\n')
185              {
186                  str[l] = '\0';
187                  break;
188              }
189
190              l++;
191          }
192
193          if (strcmp(str, inet_ntoa(cdata->saddr.sin_addr)) == 0)
194          {
195              printf("connected :%s\n", str);
196
197              refusecheck = 1;
198
199              /* (3)threadfunc()の処理を新スレッドとして実行 */
200              if (pthread_create(&th, NULL, threadfunc, cdata) != 0)
201              {
202                  perror("pthread_create");
203                  return 1;
204              }
205
206              /* (4)親スレッド側で子スレッドをdetach */
207              if (pthread_detach(th) != 0)
208              {
209                  perror("pthread_detach");
210                  return 1;
211              }
212

```

```

213             fclose(fp);
214
215             break;
216         }
217     }
218
219     if (refusecheck == 0)
220     {
221
222         printf("%s\n", inet_ntoa(cdata->saddr.sin_addr));
223
224         puts("許可されていないIPアドレスからのアクセス");
225
226         if (pthread_create(&th, NULL, refusefunc, cdata) != 0)
227         {
228             perror("pthread_create");
229             return 1;
230         }
231
232         /* (4)親スレッド側で子スレッドをdetach */
233         if (pthread_detach(th) != 0)
234         {
235             perror("pthread_detach");
236             return 1;
237         }
238     }
239 }
240
241 /* (8) */
242 /* listen するsocketの終了 */
243 if (close(sock0) != 0)
244 {
245     perror("close");
246     return 1;
247 }
248
249 return 0;
250 }

```

---

## 5.2 クライアント側

Listing 3 client

---

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>

```

```

5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <netinet/in.h>
8 #include <arpa/inet.h>
9 #include <errno.h>
10 #include <netdb.h>
11
12 int main(int argc, char *argv[])
13 {
14     struct sockaddr_in server;
15     struct addrinfo hints, *res;
16     struct in_addr addr;
17     int sock;
18     char buf[1024];
19     int n;
20     int errocheck;
21     int portnum;
22     char ipadd[16];
23
24     portnum = atoi(argv[2]);
25
26     memset(&hints, 0, sizeof(hints));
27     hints.ai_socktype = SOCK_STREAM;
28     hints.ai_family = AF_INET;
29     if ((errocheck = getaddrinfo(argv[1], NULL, &hints, &res)) != 0)
30     {
31         printf("error %d\n", errocheck);
32         return 1;
33     }
34
35     addr.s_addr = ((struct sockaddr_in *)(res->ai_addr))->sin_addr.s_addr;
36     inet_ntop(AF_INET, &addr, ipadd, sizeof(ipadd));
37     printf("ip address : %s\n", ipadd);
38
39     /* ソケットの作成 */
40     sock = socket(AF_INET, SOCK_STREAM, 0);
41     if (sock < 0)
42     {
43         perror("socket");
44         printf("%d\n", errno);
45         return 1;
46     }
47
48     /* 接続先指定用構造体の準備 */
49     server.sin_family = AF_INET;
50     server.sin_port = htons(portnum);
51

```

```

52  /* 127.0.0.1はlocalhost */
53  inet_pton(AF_INET, ipadd, &server.sin_addr.s_addr);
54
55  /* サーバに接続 */
56  errocheck = connect(sock, (struct sockaddr *)&server, sizeof(server));
57  if (errocheck < 0)
58  {
59      perror("connect");
60      printf("%d\n", errno);
61      return 1;
62  }
63
64  read(sock, buf, sizeof(buf));
65
66  if (strcmp(buf, "refused") == 0)
67  {
68      puts("connection refused");
69
70      close(sock);
71
72      freeaddrinfo(res);
73
74      return 0;
75  }
76  else
77  {
78      printf("%s\n", buf);
79
80      printf("message >");
81
82      fgets(buf, 1024, stdin);
83
84      write(sock, buf, sizeof(buf));
85
86      read(sock, buf, sizeof(buf));
87
88      printf("%s\n", buf);
89
90      /* socketの終了 */
91      close(sock);
92
93      freeaddrinfo(res);
94
95      return 0;
96  }
97 }

```

---